VŠB – technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Grafický nástroj pro modelování schémat a diagramů
Graphic Tools for Scheme and Diagram Modeling

2010 / 2011 Ivan Rožnovský

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně.

Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě dne 15. srpna 2011

….....

Ivan Rožnovský

Poděkování

Rád bych vyjádřil své poděkování především svému vedoucímu mé bakalářské práce panu Ing. Radoslavovi Fasugovi, Ph.D. za rady při vývoji aplikace a psaní této práce. Dále bych rád poděkoval své rodině a přátelům za podporu při studiu. Děkuji.

Abstrakt

Cílem práce je vytvoření aplikace, která bude napomáhat k ověřování znalostí studentů. Aplikace bude sloužit k testování znalostí v oblasti sestavování nejrůznějších schémat. Aplikace umožní studentům vytvářet schémata, podle kantorem dříve zadaných podmínek, tato schémata pak aplikace porovná se zadaným vzorem a vyhodnotí shodu. Do aplikace bude vstupovat učitel jako zadavatel schémat a tvůrce testů a student jako testovaná osoba. Hlavní části aplikace jsou implementovány jako applety v jazyce Java za využití knihovny JGraph, tyto applety jsou integrovány do webové stránky, která slouží jako ovládací panel celého systému. Data jsou ukládaná v databázi, systém funguje na principu klient-server.

Klíčová slova

Applet, Java, schéma, JGraph, PHP, http, testování studentů, databáze, XML

Abstract

The goal is to create an application that will help to verify students' knowledge. The application is used to test knowledge of drawing up various schemes. The application allows students to create diagrams, according to previously cantor set conditions, these schemes then the application compares the specified model and evaluate compliance. The application will enter as a teacher creator of test patterns and the student as the person being tested. The main part of the application are implemented as Java apletts using JGraph library, these applets are integrated into a Web page that serves as the control panel of the system. The data are stored in the database, the system works on a client-server.

Key Words

Applet, Java, Scheme, JGraph, PHP, http, testing of students, database, XML

Obsah

1. Úvod

Cílem této práce je vytvořit univerzální systém pro ověřování znalostí studentů v oblasti tvorby nejrůznějších schémat. Systém je možné využít v předmětech zabývajících se sestavováním elektronických obvodů, logických obvodů, vývojových diagramů, ale i mnohých dalších schémat, které vycházejí z teorie grafů. Kantor, který chce provádět testování, musí nejdříve specifikovat knihovnu možných součástí, případně použít některou již specifikovanou. K této knihovně připojí kritéria pro hodnocení studentova výsledku. Po vytvoření takovéto knihovny může již kantor, sestavit referenční schéma, což je v podstatě správné řešení zadaného problému. Na základě takto vytvořeného referenčního schématu, pak nástroj pro testování studenta vytvoří sadu součástí, které ve svém řešení smí student použít. Student po vytvoření své verze schématu odešle řešení, výsledek je vyhodnocen a odeslán do databáze. Učitel ve své části systému, může tyto výsledky spravovat.

Hlavní informační systém je vytvořen jako webová stránka pomocí technologií HTML a PHP. Jednotlivé části aplikace jsou tvořeny jako Applety v jazyce Java. Celý systém používá pro ukládání dat databázi a funguje na principu klient-server.

1.1. Struktura dokumentu

Tato bakalářská práce je rozdělená do několika kapitol, které popisují jednotlivé části vypracování. Po tomto úvodu následuje teoretické připomenutí implementované problematiky, teorie grafů a seznámení s použitou knihovnou JGraph. V další kapitole bude čtenář stručně seznámen se strukturou aplikace. Čtvrtá kapitola popisuje hlavní části aplikace, což jsou applety v jazyce Java. Následuje kapitola zabývající se možnostmi porovnávání schémat. Další kapitola popisuje webové rozhraní, které je základem celé implementace. Kapitola sedmá se zabývá tím, čím je specifikován vrchol a hrana v aplikace. Následuje kapitola o způsobu ukládání dat do databáze. Předposlední kapitolou je zmínka o hardwarových a softwarových požadavcích aplikace. Celou práci uzavírá závěr.

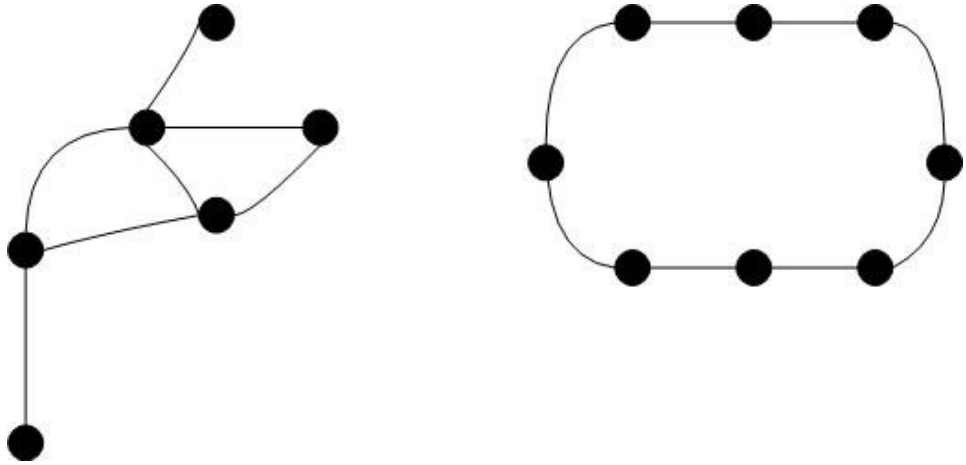
2. Teoretický úvod

Schémata, která jsou v aplikaci sestavována, jsou prakticky grafy, tak jak je známe z teorie grafů.

2.1. Úvod do teorie grafů

2.1.1. Definice grafu

Definice grafu: Graf (rozšířeně obyčejný či jednoduchý neorientovaný graf) je uspořádaná dvojice G=(V,E), kde V je množina vrcholů a E je množina hran – množina vybraných dvouprvkových podmnožin množiny vrcholů.[1]



Obrázek 1: Příklad jednoduchých grafů

Neformálně by se dalo říct, že graf se skládá z vrcholů a z hran, které spojují dvojice vrcholů mezi sebou.

2.1.2. Některé základní typy grafů

- **orientovaný graf –** graf, u kterého jsou jednotlivým hranám přiřazeny směry
- **neorientovaný graf –** graf, u něhož nejsou jednotlivým hranám přiřazeny směry
- **kružnice –** má vrcholy spojené hranami do jednoho cyklu
- **cesta –** má všechny vrcholy pospojované za sebou hranami
- **úplný graf –** má všechny vrcholy navzájem pospojované
- **strom –** graf ve kterém máme možnost se pomocí hran dostat z každého vrcholu do

každého dalšího vrcholu a přitom graf neobsahuje kružnici

2.1.3. Analýza grafů

2.1.3.1. Souvislost grafu

Souvislý graf je takový graf, v němž můžeme mezi libovolnými dvěma vrcholy grafu najít cestu. Zjednodušeně řešeno z jednoho vrcholu do jakéhokoli druhého se dostaneme putováním mezi vrcholy po hranách.

2.1.3.2. Hledání nejkratší cesty v grafu

Chceme-li nalézt nejkratší cestu mezi dvěma vrcholy grafu, využijeme tzv. Dijkstrův algoritmus.

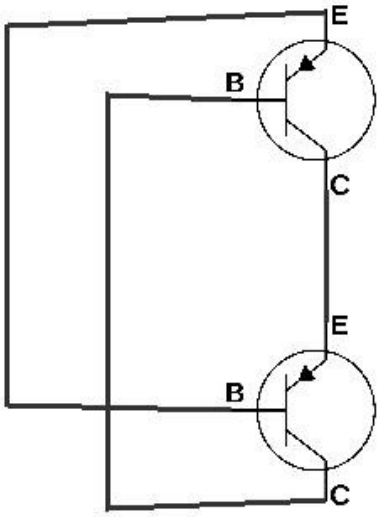
Postup algoritmu: Určíme si množinu *V* jako množinu všech vrcholů grafu a množinu *E* jako množinu všech hran grafu. Algoritmus si v každé chvíli pamatuje nejkratší cestu z úvodního vrcholu ke každému dalšímu vrcholu grafu. Na počátku je všem vrcholům přiřazena hodnota ∞, kromě úvodního vrcholu, z nějž vzdálenosti zjišťujeme, tomu přiřadíme vzdálenost 0. ∞ definuje, že je pro nás cesta k vrcholu zatím neznáma. Algoritmus si dále zřídí množiny *Z* a *N*, kdy v množině *Z* jsou již zpracované (navštívené) vrcholy a v množině N nezpracované (nenavštívené) vrcholy. Na počátku umístíme do množiny *N* všechny vrcholy. Algoritmus pracuje tak dlouho dokud množina *N* není prázdná. V každém cyklu se vrchol, který má v množině *N* nejmenší známou vzdálenost přesune do množiny *Z*. Pro každý vrchol *u*, do kterého vede cesta z právě přesunovaného vrcholu zjišťujeme zda vzdálenost přesunovaného vrcholu + vzdálenost z něj do *u* není menší, než dosud známá nejkratší vzdálenost pro *u,* pokud je menší, pak tuto hodnotu přiřadíme jako nejkratší vzdálenost do *u*. Až algoritmus skončí, má každý vrchol v množině Z přiřazenou nejkratší vzdálenost z úvodního bodu.

2.2. JGraph

JGraph je open source knihovna napsaná v jazyce Java, která slouží pro vizualizaci grafů. Grafem je v tomto případě objekt teorie grafů, nikoli koláčový graf, sloupcový graf a podobně.

Knihovna JGraph je plně kompatibilní s architekturou Java Swing.

Zatímco teorie grafů řeší jen propojení jednotlivých vrcholů hranami, JGraph navíc řeší i zobrazení jednotlivých vrcholů a hran a pomáhá tak vytvořit přehlednou vizualizaci grafu.



Obrázek 2: Příklad vizualizace za použití JGraphu

2.2.1. Základní rozhraní a metody v JGraphu

třída JGraph
JGraph je hlavní třída v JGraphu, tato třída rozšiřuje třídu javax.swing.JComponent a drží referencí na svůj model (rozhraní GraphModel), pohled (třída GraphLayoutCache) a uživatelské rozhraní (třída GraphUI). Základní struktura komponenty je založená na architektuře Swing MVC, která je zděděná z třídy JTree.

rozhraní GraphModel
GraphModel drží logickou strukturu grafu.

třída DefaultGraphModel
Základní implementace modelu grafu (rozhraní GraphModel).

třída GraphLayoutCache

Třída určuje, jak bude model interpretován a zobrazen.

rozhraní GraphCell
Rozhraní, které by měly dědit všechny vrcholy a hrany. Rozhraní GraphCell je považováno za základní pro vrcholy grafu.

rozhraní Edge
Rozhraní, které dědí z rozhraní GraphCell a slouží jako základní pro hrany grafu.

Základní metody:

getSource() - vrátí vrchol, z něhož hrana vychází

setSource(Object port) - nastaví vrchol, z něhož hrana vychází

getTarget() - vrátí vrchol, do něhož hrana směřuje

setTarget(Object port) - nastaví vrchol, do něhož hrana směřuje

rozhraní Port
Rozhraní, které dědí z rozhraní GraphCell a slouží jako základní pro porty grafu (body, které zajišťují propojení hran a vrcholů).

Základní metody:

edges() - vrátí iterátor hran, připojených k vrcholu

addEdge(Object edge) – přidá hranu do seznamu hran připojených k portu

removeEdge(Object edge) – odebere hranu ze seznamu hran připojených k portu

třída DefaultGraphCell
Základní implementace rozhraní GraphCell. Třída dědí z javax.swing.tree.DefaultMutebleTreeNode. Třída DefaultGraphCell je základní implementací vrcholu grafu v JGraphu.

třída DefaultEdge
Třída implementující rozhraní Edge a rozšiřuje DefaultGraphCell. Základní třída pro reprezentaci hrany v JGraphu.

třída DefaultPort
Třída implementující rozhraní Port a rozšiřuje DefaultGraphCell. Základní třída pro reprezentaci portu v JGraphu.

třída CellView
Třída definující výsledné zobrazení prvku grafu.

3. Struktura aplikace

Aplikaci můžeme rozdělit na tři části. První částí jsou applety v jazyce Java, ty slouží přímo k definici jednotlivých knihoven, schémat a k jejich zobrazení. Tyto applety najdeme v aplikaci celkem čtyři, první je jako editor knihoven vrcholů a hran, druhý je editor schématu učitele, třetí editor schématu studenta a čtvrtý prostý prohlížeč hotových schémat. Implementace těchto appletů využívá knihovnu JGraph, konkrétně její verze 5.13.0.0, tato verze bylo zvolena proto, že byla aktuální v době kdy jsem začínal tuto práci dělat. Applety jsou rozebrány v kapitole 4.

Druhá část je webové rozhraní. Jedná se o část, která je v podstatě řídícím centrem celé aplikace. Slouží k vytváření a správě informací o knihovnách, schématech, studentech, pracovních skupinách, testech, výsledcích. Webové rozhraní je tvořeno jako webová stránka v jazyce HTML z využitím serverového skriptovacího jazyku php. O webovém rozhraní se čtenář více dočte v kapitole 6.

Poslední částí je databáze, v níž jsou uložena všechna data. S databází komunikuje webové rozhraní a prostřednictvím php skriptů i applety. Můžeme jí tedy považovat za jakési jádro aplikace. Databázi a komunikaci s ní se věnuje kapitola 8.

4. Applety

Části aplikace jako editor knihoven, učitelský editor schémat, studentský editor schémat a prohlížeč hotových schémat jsou realizovány jako applety v jazyce Java. V další části textu se budu věnovat jednotlivým částem.

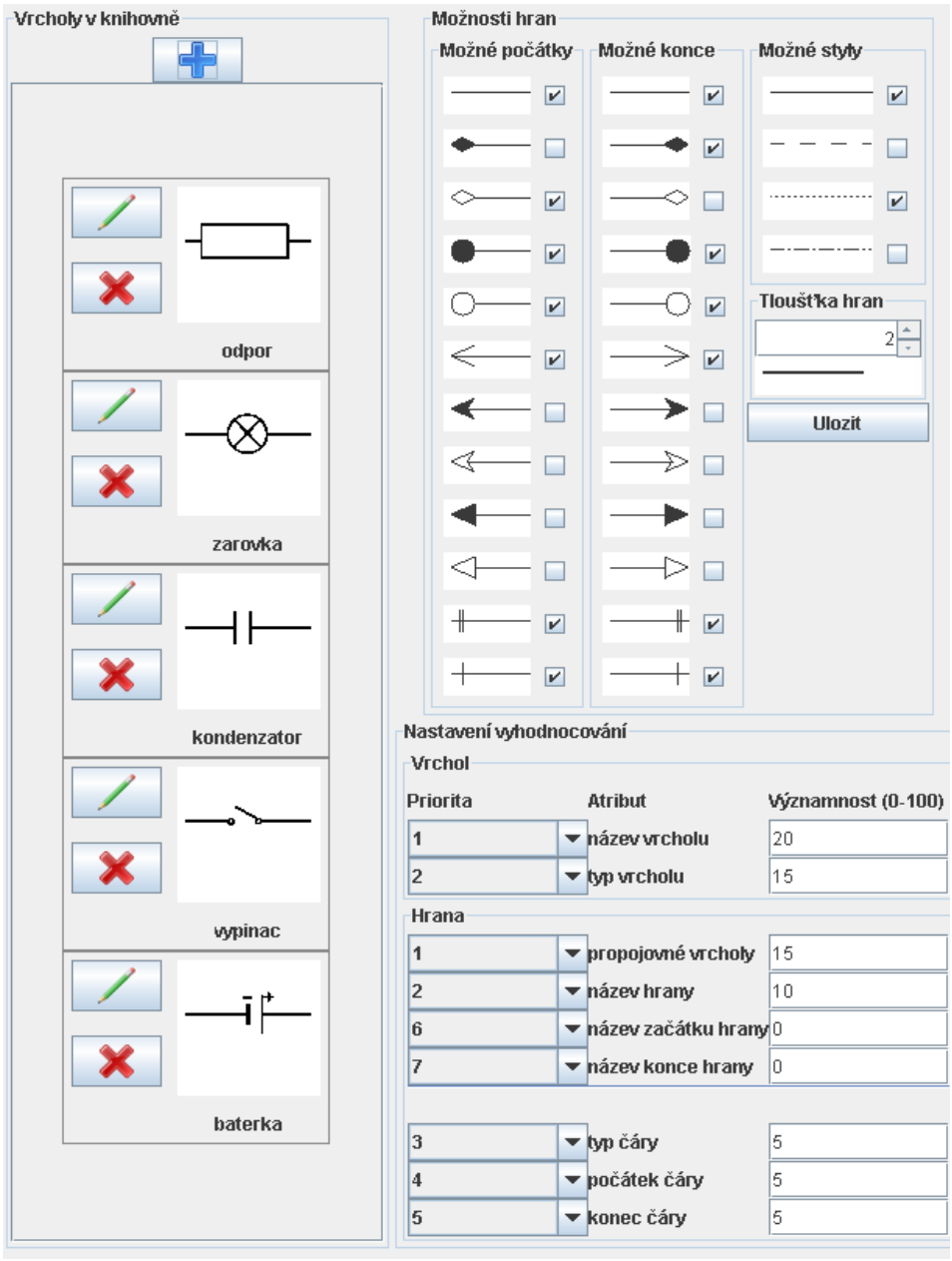
4.1. Umístění appletu do HTML kódu

Applet je zaveden do webové stránky pomocí párového tagu jazyka HTML <APPLET>. Jako atributy počáteční značky jsou uvedeny cesta ke zdrojovému kódu appletu, šířka appletu a výška appletu. Mezi úvodním a koncovým tagem <APPLET> se umisťují značky

*<param name="nazev_parametru" value="*hodnota_parametru*">*, které specifikují různé vlastnosti nutné pro běh appletu, například id schématu, aby applet mohl načíst z databáze správné schéma. Jednotlivé applety přijímají různé parametry.

4.2. Editor knihoven a pravidel hodnocení

První applet je editor knihoven a pravidel hodnocení. S touto částí aplikace pracuje výhradně učitel a navrhuje v ní jednotlivé typy vrcholů, možnosti hran a pravidla pro porovnávání řešení. Zadáním těchto parametrů dojde k vytvoření knihovny, která je dále využívaná pro definici samotného schématu.



Obrázek 3: Ukázka editoru knihoven a vyhodnocování

4.2.1. Panel vrcholů

Panel vrcholů zobrazuje jednotlivé objekty (vrcholy), které jsou v knihovně přidány. Jednotlivé vrcholy je možné mazat a upravovat pomocí tlačítek vedle nich. Kliknutím na horní tlačítko je možné přidat nový vrchol. Po kliknutí se zobrazí dialog, v němž se specifikují vlastnosti vrcholu.

Jednotlivému vrcholu nastavujeme jedinečný název typu, který ho identifikuje mezi všemi vrcholy v knihovně. Dále zadáme adresu obrázku, který objekt symbolizuje a načteme jej. Po načtení obrázku můžeme nastavit jednotlivé body, kde se bude dát připojit hrana (vstupy a výstupy objektu), můžeme upravit velikost objektu.

4.2.2. Natavení hran

V levé části editoru knihoven najdeme nastavení možností hran, můžeme vybrat jednotlivé typy začátků a konců hran, styly čáry hrany a tloušťku hran. Vybrané možnosti pak bude možné použít při tvorbě schématu.

4.2.3. Nastavení vyhodnocování

Pod nastavením hran nalezneme nastavení pravidel pro vyhodnocování (porovnávaní) schémat. Aplikace schémata porovnává pomocí objektového přístupu, více se o tomto porovnávání zmiňuji v kapitole 5.1.1. V této části nastavíme priority porovnávaní vrcholů a propojujících hran, a také významnost jednotlivých vlastností v konečném hodnocení.

4.2.4. Uložení knihovny do databáze

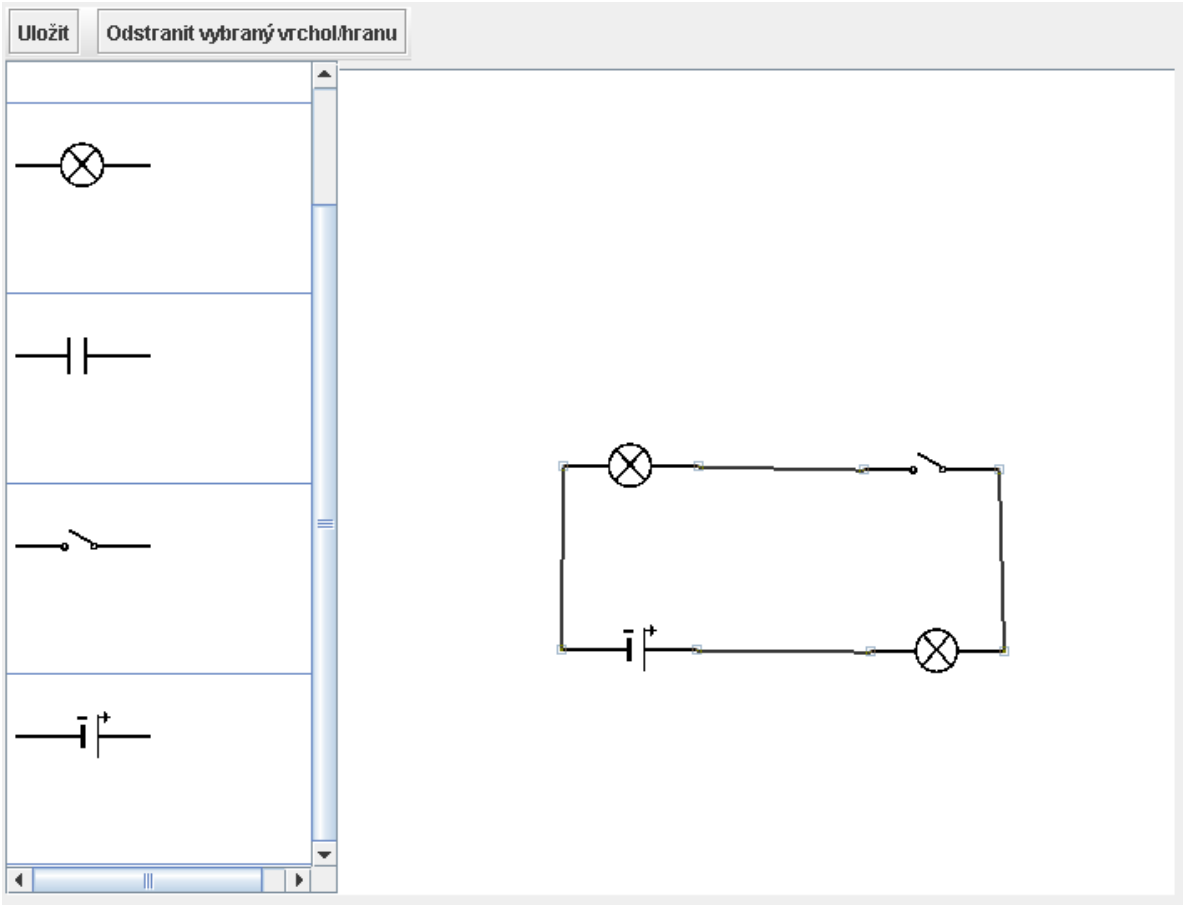
Po dokončení úprav knihovny je nutné aktuální stav knihovny uložit do databáze, to provedeme stisknutím tlačítka Uložit.

4.2.5. Zavedení editoru knihoven do HTML kódu

Applet editoru knihoven je zaveden do kódu dříve uvedeným způsobem. Přijímá jeden parametr, jehož jméno je **id** a jeho hodnota označuje identifikační číslo knihovny, pod nímž je knihovna uložena v databázi.

4.3. Učitelský editor schémat

Editor schématu učitele je druhým vytvořeným appletem. Na základě vytvořené knihovny nabízí tento editor možnost tvořit libovolné schéma. Vytvořené schéma je uloženo v databázi a může být použito jako základ testu pro studenty.



Obrázek 4: Ukázka práce v učitelském editoru schémat

4.3.1. Nabídkový panel

V levé části obrazovky učitelského editoru schémat se nachází nabídkový panel, v něm jsou zobrazeny všechny vrcholy, které jsou v použité knihovně a můžeme je použít. Kliknutím na vybraný objekt a následným kliknutím na plátno vpravo umístíme objekt na plátno.

4.3.2. Hlavní plátno

Pravou část editoru zaujímá hlavní plátno na němž je tvořeno schéma. O přidávaní vrcholů byla řeč v předchozím textu. Hranu mezi objekty můžeme vytvořit tak, že klikneme na malý

čtverec, který označuje místo, kde je přípojný bod objektu (port) z tohoto bodu pak za stálého stisku levého tlačítka myší táhneme hranu k druhému přípojnému bodu.

4.3.2.1 Nastavení vlastností vrcholu

Kliknutím pravým tlačítkem na konkrétní vrchol nám vyskočí tlačítko edit. Po kliknutí na něj se nám zobrazí dialog nastavení vlastností vrcholu.

Obrázek 5: Nastavení vlastností vrcholu - učitelský editor

V tomto dialogu nastavíme jméno vrcholu a také zda má být vrchol zobrazen studentovi v testu od začátku na plátně a zda je vrchol pravý nebo jen falešný pro zmatení studenta.

4.3.2.2 Nastavení vlastností hrany

Kliknutím pravým tlačítkem na konkrétní hranu nám vyskočí tlačítko edit. Po kliknutí na něj se uživateli zobrazí dialog pro nastavení vlastností hrany.

Obrázek 6: Nastavení vlastností hrany - učitelský editor

V zobrazeném dialogu se nastaví styl počátku a konce hrany, styl čáry hrany, název hrany a opět vlastnosti falešná hrana a hrana zobrazená studentovi od počátku.

4.3.3. Ukončení práce na schématu

Schéma učitel po musí po dokončení uložit. To provede stisknutím tlačítka uložit v horní části appletu

4.3.4. Zavedení učitelského editoru schémat do HTML kódu

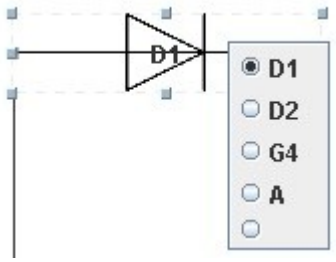
Applet editoru učitelského editoru schémat je zaveden do kódu způsobem uvedeným v 4.1. Přijímá dva parametry, **idl** jehož hodnota označuje id knihovny v databázi a **idt** jehož hodnota označuje id učitelského schématu v databázi.

4.4. Studentský editor schémat

Dalším appletem je editor schémat pro studenta. Na první pohled vypadá podobně jako editor učitelův. Vlevo vidíme nabídku vrcholů, jsou tam uvedeny ty typy vrcholu, které ve svém schématu umístil učitel, včetně falešných. Vpravo je pak hlavní plátno na němž se schéma tvoří. Na počátku jsou na něm zobrazeny vrcholy a hrany, které byly učitelem zvoleny jako viditelné na začátku.

4.4.1. Nastavení názvu vrcholu

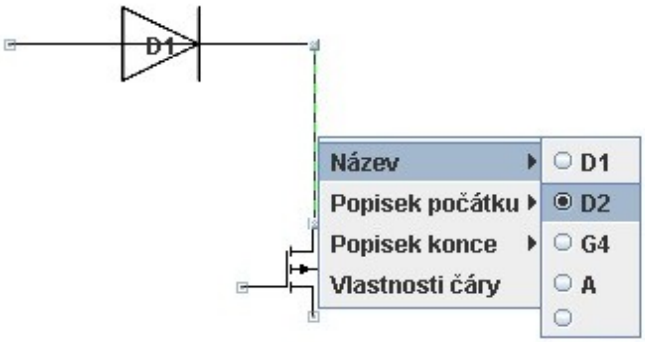
Po kliknutí pravým tlačítkem na konkrétní vrchol je studentovi zobrazena nabídka názvů vrcholu, jsou zde uvedeny všechny názvy, které učitel použil ve svém schématu.



Obrázek 7: Nastavení názvu vrcholu - student

4.4.2. Nastavení vlastností hrany

Po kliknutí pravým tlačítkem na konkrétní hranu je zobrazena nabídka nastavení vlastností této hrany. Studentovi jsou nabídnuty všechny možnosti, které použil ve svém schématu učitel.



Obrázek 8: Nastavení vlastností hrany - student

4.4.3. Ukončení práce a vyhodnocení

Po ukončení tvorby stiskne student tlačítko hotovo, applet ihned porovná schéma s učitelovým schématem a vrátí výsledek testu. Výsledné schéma i výsledek uloží do databáze.

4.4.4. Zavedení studentského editoru schémat do HTML kódu

Applet studentského editoru schémat je zaveden do kódu způsobem uvedeným v 4.1. Přijímá dva parametry, **idt** jehož hodnota označuje id vybraného testu v databázi, **login** jehož hodnota určuje studenta který absolvuje test.

4.5. Prohlížeč hotových schémat

Posledním appletem je prohlížeč, který umožňuje učiteli zobrazit výsledek vytvořený studentem. V tomto appletu nelze schéma nijak upravovat, pouze jej prohlížet.

4.5.1. Zavedení prohlížeče schémat do HTML kódu

Applet prohlížeče schémat je zaveden do kódu způsobem uvedeným v 4.1. Přijímá dva parametry, **idTest** jehož hodnota označuje id vybraného testu v databázi, **login** jehož hodnota určuje studenta který absolvoval test.

5. Porovnání schémat

5.1. Možnosti porovnávání schémat

Schémata studenta a učitele je možno porovnávat několika způsoby. Jsou rozebrány níže.

5.1.1. Objektové porovnání

Dochází k porovnávání objektů na učitelském i studentském schématu. Výsledkem porovnání je tzv. procento podobnosti, kdy 100% je dosaženo tehdy, je-li schéma studenta shodné se schématem učitele (obsahuje stejné vrcholy a stejné hrany). Pro posouzení podobnosti obou schémat porovnáváme vrcholy a hrany v jednotlivých schématech, vrcholy jsou popsány dvěma atributy a to typem vrcholu a názvem vrcholu, hrany pak můžeme rozlišovat podle názvu, typu čáry, propojovaných objektů, jednotlivých zakončení čáry a názvů konců hran. Při zadání hodnocení je na nás určení priority jednotlivých atributů a jejich významnost v celkovém hodnocení. U některých typů schémat můžeme některé atributy chápat jako bezpředmětné, na ty při porovnávání nebereme zřetel (jejich významnost je 0).

5.1.1.1. Postup porovnání

Porovnání provádíme ve třech cyklech:

- **zjišťujeme co má student společného s učitelem**

  V tomto cyklu se pokoušíme v studentském schématu najít stejné objekty jako jsou v učitelském schématu. Začneme vyhledáváním vrcholů, vybereme vrchol z učitelského schématu a pokoušíme se jej najít ve studentském podle předem specifikovaných pravidel pro vyhledávání (podle priority jednotlivých atributů), u vrcholů máme specifikovány dva atributy – název a typ. Logičtější je nejprve hledat zda ve studentském schématu najdeme vrchol se stejným názvem, pokud ano, pak zjišťovat zda se jedná i o správný typ vrcholu. Samozřejmě může být priorita nastavena i opačně, pak nejprve hledáme vrchol stejného typu a až jej najdeme, tak kontrolujeme jeho název, je zřejmé že v tomto případě můžeme najít více než jeden objekt stejného typu a tudíž je musíme porovnat všechny. Po vyhledání všech vrcholů se pustíme do vyhledávání hran. U hran máme více atributů podle nichž se dá vyhledávat stejná hrana v studentském a učitelském modulu. Důležité atributy, které by měly mít nejvyšší prioritu jsou: název, propojované objekty, název levé strany hrany a název pravé strany hrany. Dalšími atributy jsou typ čáry, levý konec čáry, pravý konec čáry, případně oba konce hodnotíme současně. Jak u vrcholů, tak u hran udělujeme body jednotlivým shodným atributům podle nastavené významnosti jednotlivých atributů.
  Příklad: pro název vrcholu máme nastavenou významnost 3 a vyšší prioritu než pro typ vrcholu, pro typ vrcholu máme danou významnost 2. V případě že nalezneme vrchol

stejného názvu i stejného typu dostane za něj student 5 bodů (3 za název + 2 za typ), nalezneme-li vrchol stejného názvu, ale jiného typu obdrží student jen 3 body za správný název, v případě že nenalezneme vrchol stejného názvu obdrží za tento vrchol student 0 bodů.

Na závěr sečteme všechny body za všechny vrcholy a hrany. Na úplný závěr (po případném provedení dalších dvou cyklů) vydělíme celkový možný počet bodů udělenými body a tak získáme procento podobnosti.

- **zjišťujeme co studentovi chybí od učitele**

  V tomto cyklu zjišťujeme co studentovi chybí oproti učiteli. Tento krok jsme v podstatě vyřešili již v prvním cyklu, tím že jsme mu neudělili body za ty vrcholy a hrany které mu chyběly, či které nebyly zcela správné.

- **zjišťujeme co studentovi přebývá oproti učiteli**

  Tento cyklus se provádí v případě, že chceme výsledek prvního cyklu degradovat na základě přebývajících či špatně označených vrcholů a hran. Vyhledáme ve studentově schématu vrcholy a hrany, které nejsou v učitelově a podle jejich významnosti odečteme body od výsledku prvního cyklu. Máme-li nastavenou degradaci do nuly, můžeme pak v případě dosáhnutí nuly tento cyklus ukončit a vrátit výsledek 0%.

  Příklad: v prvním cyklu student obdržel 12 bodů, v tomto cyklu pak zjistíme že mu ve schématu přebývá jedna hrana, která má po sečtení všech atributů významnost 7, odečteme tedy od 12 bodů těchto 7 a výsledkem je 5 bodů pro studenta.

  Z výsledných bodů a maximálního možného počtu bodů na závěr vypočteme procento podobnosti.

5.1.1.2. Způsoby vyhodnocení

- pouze shodné prvky – při tomto porovnání nebereme ohled na objekty a hrany které student přidal navíc oproti učiteli, započítáme body za prvky, které má student shodné s učitelem
- degradace výsledků – při tomto porovnání nejprve zjistíme co má student společného s učitelem a za to udělíme body, dále pak na základě přebývajících nebo špatně ohodnocených vrcholů a hran body odečítáme. U tohoto hodnocení lze rozlišit dvě možnosti hodnocení
  - do nuly – měl-li by výsledek být záporný, zaokrouhlíme jej na nulu
  - do záporu – procento podobnosti může nabývat záporných hodnot

5.1.2. Hodnotové (logické) porovnání

Bereme zřetel na logiku, funkčnost řešení. Není podstatné zda schéma studenta vypadá stejně jako schéma učitele, ale zda plní stejnou funkci.

5.1.3. Poziční porovnání

Student obdrží prázdnou plochu, případně se zobrazením obrysů. Příkladem může být slepá mapa Evropy, kdy student obdrží obrázek s obrysem Evropy a jednotlivé státy a jeho úkolem je přetažením umístit státy do správné pozice, hodnocení je určeno na základě rozdílu mezi umístěním studenta a správným řešením. Obměnou může být opět mapa s obrysem Evropy a úkolem studenta je kliknout co nejblíže městu, jehož název je mu nabídnut.

5.2. Třída ModelComparator

V mém řešení je použita metoda objektového porovnání, popsána v kapitole 5.1.1.. Porovnání řeší metoda ModelComparator.

5.2.1. Konstruktor

Celé porovnání se řeší v konstruktoru

```
public ModelComparator(GraphModel teacher, GraphModel student, CompareRules rules)
```

Parametry konstruktoru jsou:

- **teacher** – objekt třídy GraphModel, reprezentující model učitelského schématu
- **student** – objekt třídy GraphModel, reprezentující model studentského schématu
- **rules** – objekt třídy CompareRules, reprezentující pravidla pro porovnání

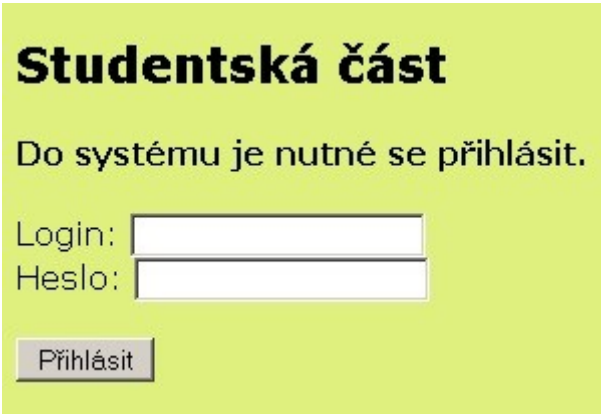
Metoda `double percentResult()` zavolaná na vytvořený objekt typu ModelComparator vrací procentuální hodnotu podobnosti.

6. Webové rozhraní

Webové rozhraní slouží, jako ovládací panel celého systému. Je rozděleno na dvě části: studentskou a učitelskou. V učitelské části je možné spravovat studenty, skupiny studentů, knihovny, schémata, testy a výsledky testů. Student může ve své části absolvovat aktuálně otevřené testy a můžou si prohlížet výsledky svých absolvovaných testů. Obě části budou detailněji popsány níže. Pro tvorbu a testování je toto webové rozhraní umístěno na mých osobních stránkách *http://acter0.cz*.

6.1. Přihlášení

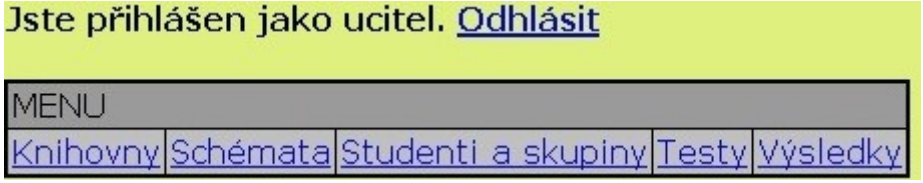
Pro použití systému je nutné se přihlásit. Jednotliví přidaní studenti dostanou na úvod heslo stejné jako login, po přihlášení se jej mohou změnit. Stejným způsobem se přihlašují učitelé, kteří jsou do systému přidáni pomocí jiného učitele.



Obrázek 9: Formulář pro přihlášení

6.2. Učitelská část

Po přihlášení do systému se učiteli zobrazí hlavní menu, z něhož má přístup ke všem částem svého webového rozhraní.



Obrázek 10: Menu webového rozhraní učitele

6.2.1. Sekce Knihovny

V této sekci najdeme seznam všech vytvořených knihoven, můžeme zde taky vytvořit novou knihovnu. Kliknutím na odkaz *upravit* v seznamu knihoven přejdeme na editaci vybrané knihovny.

6.2.2. Sekce Schémata

Pod touto volbou nalezneme seznam všech vytvořených schémat, můžeme tady také vytvořit nové schéma. Kliknutím na odkaz *upravit* v seznamu schémat se dostaneme k editaci daného schématu.

6.2.3. Sekce Studenti a skupiny

V této sekci máme možnost přidávat studenty a studentské skupiny, tyto zde můžeme také prohlížet a také zde můžeme studenty vkládat do skupin. V této sekci nalezneme také odkaz na přidání nového učitele.

6.2.4. Sekce Testy

V části testy můžeme zobrazit seznam vytvořených testů, a také založit nový test tím, že specifikujeme skupinu pro níž je test určen, schéma které se má řešit, název testu, textové zadaní a časové rozmezí v němž je možno test složit.

6.2.5. Sekce Výsledky

Zde najdeme výsledky absolvovaných testů. Můžeme zobrazit dosažené výsledky v procentech a také výsledné schéma.

6.3. Studentská část

Studentovi se po přihlášení zobrazí seznam testů, které může právě absolvovat. Každý test může student absolvovat pouze jednou. Pod tímto seznamem nalezne student seznam výsledků předchozích absolvovaných testů. Student spustí test klepnutím na odkaz absolvovat u daného testo. Pokud by se student pokoušel některý test absolvovat podruhé, nebude mu to umožněno. Poté co už student jednou spustil řešení testu, nemůže jej znovu spustit, i kdyby neodeslal výsledek do databáze.

7. Specifikace vrcholu a hrany

Každý vrchol reprezentovaný třídou DefaultGraphCell a každá hrana reprezentovaná třídou DefaultEdge si nese reference na dva důležité objekty. Jsou to objekt třídy Map, který specifikuje vlastnosti daného elementu grafu a také libovolný uživatelský objekt, který si daný vrchol či hrana nese s sebou.

7.1. Specifikace vrcholu

Jako uživatelský objekt je v mé práci přiřazena každému vrcholu instance třídy VertexAttributes. Tato třída definuje několik důležitých vlastností vrcholu. Základní metody třídy jsou popsány níže.

- setType(String type) – nastavení pojmenování daného typu vrcholu, pro jednoznačnou identifikaci typu.
- setName(String name) – nastavení jména konkrétního vrcholu, pro identifikaci vrcholu ve schématu.
- setIncorrect(boolean incorrect) – určení zda vrchol je falešným vrcholem
- setDisplayed(boolean displayed) – určení zda vrchol je zobrazen studentovi na počátku testu
- setImageURL(String imageURL) – nastavení cesty k obrázku reprezentujícím vrchol

Další vlastnosti se nastavují přímo na poli atributů vrcholu. Využívá se k tomu třídy GraphConstants, která obsahuje metody na nastavení jednotlivých vlastností. K hlavním metodám této třídy pro práci s vrcholy patří následující metody.

- setBounds – nastavení rozměrů a polohy vrcholu na plátně
- setIcon – nastavení podoby vrcholu, přiřazení obrázku
- setValue – nastavení uživatelského objektu (výše zmíněné instance VertexAttributes)

Všechny výše uvedené metody mají samozřejmě i své protějšky, v nichž je slovo set nahrazeno slovem get respektive is.

Důležitou součástí vrcholů jsou porty, jsou to body kde se můžou na vrchol napojovat hrany. Vytvoření takovéhoto bodu proběhne vytvořením nové instance třídy DefaultPort. Přiřazení portu k vrcholu proběhne metodou setParent(vrchol) zavolanou na instanci DefaultPort. Pro nastavení pozice bodu na vrcholu je využita statická metoda setOffset třídy GraphConstants.

7.2. Specifikace hrany

U hran je uživatelským objektem instance třídy EdgeAttributes. Metody jsou obdobné jako u VertexAttributes, pouze zde chybí metoda setImageURL, která není nutná, jelikož hranu nereprezentuje žádný obrázek. Také zde není metoda setType, jelikož typ hrany je určen jednotlivými vlastnostmi (typ čáry, zakončení čáry) hrany.

K hlavním metodám třídy GraphConstants pro práci s hranami patří níže uvedené metody.

- setLineBegin, setLineEnd – nastavení počátku a konce hrany
- setBeginFill, setEndFill – nastavení výplně počátku a konce hrany
- setBeginSize, setEndSize – nastavení velikosti zakončení hrany
- setValue – přiřazení uživatelského objektu hraně (v našem případě EdgeAttributes)

Nastavení bodů mezi nimiž je hrana se provádí voláním metody setSource (výchozí bod) a setTarget (cílový bod) přímo na objekt hrany.

8. Způsob ukládání dat

Data jsou ukládaná do databáze prostřednictvím PHP skriptů. Data jako informace o studentech, testech, knihovnách a schématech jsou ukládána přímo z informačního systému PHP, schémata a knihovny z appletů jsou ukládány asynchronním způsobem, kdy applet pošle dotaz na php skript. Tento dotaz obsahuje jako svou POST informaci i XML, které se má do databáze vložit. Pro ukládání schémat do databáze jsem navrhl vlastní univerzální XML schéma.

8.1. Návrh univerzálního XML schématu

Schémata, která v aplikaci vytvoří student či kantor je nutné ukládat do databáze. Pro toto ukládání jsem navrhl XML schéma, které bude níže vysvětleno.

K převodu do modelu grafu do tohoto XML a opačně slouží třídy ModelXMLDecoder a ModelXMLEncoder.

Nejprve uvedu příklad a dále uvedu význam jednotlivých tagů.

```
<scheme>
  <vertex>
    <type>dioda</type>
    <name>D1</name>
    <bounds>
      <height>40.0</height>
      <width>155.0</width>
      <x>326.0</x>
      <y>73.0</y>
    </bounds>
    <displayed>true</displayed>
    <incorrect>false</incorrect>
    <imageURL>http://roz172.ihukvaldy.cz/dioda.png</imageURL>
    <port>
      <id>0</id>
      <offset>
        <x>10.0</x>
        <y>520.0</y>
      </offset>
    </port>
    <port>
      <id>1</id>
      <offset>
        <x>1000.0</x>
        <y>520.0</y>
      </offset>
```

```
    </port>
  </vertex>
  <edge>
    <source>1</source>
    <target>7</target>
    <properties>
      <linewidth>1.0</linewidth>
      <lineEnd>0</lineEnd>
      <lineBegin>0</lineBegin>
      <endSize>20</endSize>
      <beginSize>20</beginSize>
      <endFill>false</endFill>
      <beginFill>false</beginFill>
      <name></name>
      <displayed>false</displayed>
      <incorrect>false</incorrect>
      <dashPattern>
      </dashPattern>
    </properties>
  </edge>
</scheme>
```

Následuje vysvětlení významu jednotlivých tagů.

- tag <scheme> obaluje celé schéma v rámci XML
- tag <vertex> tag označující jednotlivý objekt schématu (vrchol v grafu)
- tag <edge> tag označující jednotlivou hranu spojující vrcholy
- tag <port> specifikuje jednotlivý přípojný bod na daném prvku
- tag <name> pojmenování daného vrcholu či hrany
- tag <type> pojmenování typu vrcholu
- tag <bounds> specifikuje rozměry a umístění daného vrcholu
- tag <displayed> specifikuje zda má být daný vrchol či hrana zobrazen studentovi na začátku
- tag <incorrect> specifikuje zda se má zadaný vrchol či hrana vyskytovat i ve schématu vytvořeném studentem, tzn. zda není umístěna jako falešná hrana pro zmatení
- tag <offset> v tomto tagu specifikujeme umístění přípojného bodu v rámci vrcholu
- tag <imageURL> specifikuje URL adresu na níž je umístěn obrázek, který ztvárňuje vrchol
- tag <source> udává id portu z něhož vychází specifikovaná hrana
- tag <target> udává id portu do nějž míří specifikovaná hrana

- tag <properties> obaluje vlastnosti hrany
  - tag <linewidth> určení šířky hrany
  - tag <lineEnd> a <lineBegin> určení typu počátku a konce hrany
  - tag <endSize> a <beginSize> specifikuje velikosti počátku a konce hrany
  - tag <endFill> a <beginFill> určuje zda má být počátek či zakončení hrany být vyplněno
  - tag <dashPattern> určuje styl čára – plná, přerušovaná, apod.

8.2. Databáze

Data jsou ukládány do databáze MySQL. Jelikož přímá komunikace appletu s databází není možná, je komunikace řešena prostřednictvím php skriptů. Applet vyšle POST požadavek na php skript na serveru, skript se spojí s databází a následně uloží data do databáze nebo data z databáze zašle pomocí http odpovědi zpátky appletu. Knihovny, schémata a porovnávací pravidla jsou přenášeny ve formátu XML.

8.2.1. Struktura databáze

Databázi tvoří celkem osm tabulek. Z důvodu kolizí v databázi na níž jsem systém implementoval jsou tabulky pojmenovány s předponou *bc_*. Kompletní datový model je k dispozici jako příloha na CD přiloženém k této práci. Tady budou popsány jen nejdůležitější tabulky.

Jednou z nich je tabulka *bc_libraries* obsahuje vytvořené knihovny. Každý záznam v této tabulce obsahuje jednoznačný identifikátor záznamu *id_lib*, další sloupce v této tabulce jsou *code,* což je samotný XML kód knihovny, *name,* specifikující jméno knihovny a *author,* což je login autora knihovny.

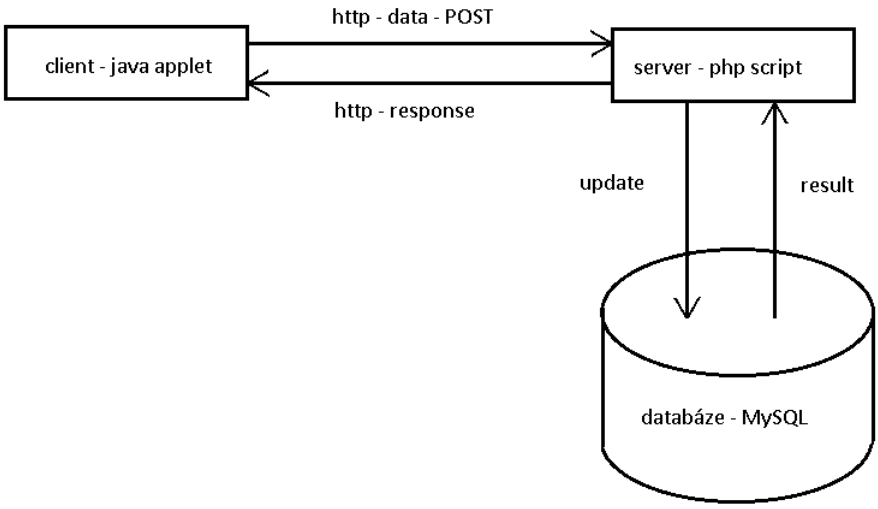
Další důležitou tabulkou je tabulka *bc_schemates*, ve které jsou uloženy jednotlivé učitelem vytvořené schémata. Sloupce této tabulky jsou *id_schema*, což je primární klíč – jednoznačný identifikátor záznamu. Dalšími sloupci tabulky jsou opět *author*, *name*, které mají stejný význam jako v předchozí tabulce, dále *id_lib* což je cizí klíč – identifikátor knihovny použité na tvorbu tohoto schématu a sloupec *code* obsahující samotné schéma ve formátu XML.

Tabulka *bc_tests* obsahuje jednoznačný identifikátor testu *id_test*, cizí klíče *id_schema*, který definuje identifikátor schématu použitého v testu a *id_wg*, který specifikuje skupinu studentů pro níž je test určen. Tabulka dále obsahuje sloupce *date_from* a *date_to*, které určují časové rozpětí, kdy je možné test složit, sloupec *task* obsahující textové zadání a *name* což je jméno testu.

Zbývající tabulky jsou *bc_results* obsahující výsledky testů, *bc_teachers* obsahující přihlašovací údaje učitelů, *bc_students* obsahující údaje o studentech, *bc_workgroups* obsahující informace o pracovních skupinách studentů a tabulku *bc_students_workgroups*, jež popisuje vazby mezi studenty a skupinami.

8.2.2. Ukládání dat do databáze

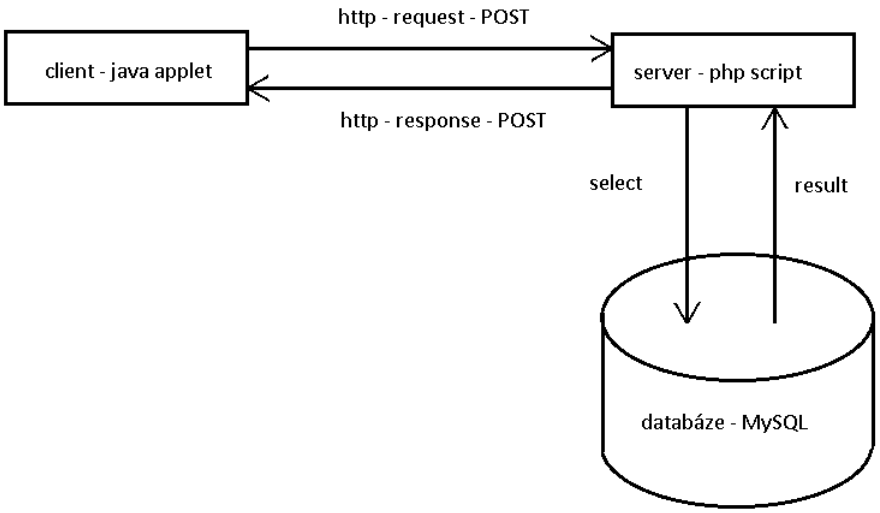
Jak již bylo zmíněno dříve, ukládání do databáze probíhá prostřednictvím php skriptu. Pro tyto účely je v řešení použito několika souborů. Pro uložení knihovny saveLibrary.php, učitelské schéma se ukládá pomocí saveTeacherScheme.php a výsledek studenta se ukládá přes soubor saveResults.php. Applet pošle na konkrétní soubor http dotaz, který pomocí POST informace předá XML data a skript je poté těmito daty aktualizuje tabulku v databázi. Schéma komunikace je na následujícím obrázku.



Obrázek 11: Komunikace appletu s databází při ukládání

8.2.3. Načítání dat z databáze

Načítání z dat z databáze probíhá obdobně jako ukládání. Na tuto komunikaci jsou tady tři soubory loadLibrary.php na načítání knihovny, loadTeacherScheme.php na načítání učitelského schématu a loadResults.php na načtení výsledků studenta. Applet odešle http požadavek na soubor, kdy mu předá informace o datech, které požaduje. Po obdržení požadavku php skript vyhledá XML data v databázi a jako POST informaci v http odpovědi je odešle appletu. Komunikace je znázorněna na následujícím obrázku.



Obrázek 12: Komunikace appletu s databází při načítání dat

9. Hardwarové a softwarové požadavky

9.1. Hardwarové požadavky

Z hlediska klientské části aplikace, by aplikace měla být bez problémů funkční na většině současných osobních počítačů. Doporučen je počítač s procesorem s frekvencí alespoň 1GHz a operační pamětí minimálně 512MB. Jistě však poběží i na některých slabších počítačích. Jelikož rozlišení většiny appletů se pohybuje kolem 900x600 pixelů, je vhodné mít monitor a grafickou kartu podporující rozlišení alespoň 1024x768 pixelů. Jelikož se klientská aplikace připojuje na server, je nutné připojení k internetu. Server na němž bude umístěn web a databáze by měl mít dostatek diskové paměti, pro uložení dostatečného množství řešení.

9.2. Softwarové požadavky

Pro správnou funkci aplikace je nutné mít na straně serveru nainstalován webový server s podporou php a databázi MySQL. Po mírných úpravách php skriptů by bylo možné provozovat i na jiném typu databáze.

Na straně klienta je nezbytná instalace běhového prostředí Javy v minimální verzi 1.6. Také webový prohlížeč musí podporovat Javu. Vzhledem k tomu, že prostředí Javy je nezávislé na systému neměl by být problémem provoz ani na systému Windows ani v systému Linuxu. Aplikaci jsem testoval na platformě Windows 7 v prohlížečích Google Chrome, Mozilla Firefox a Microsoft Internet Explorer 8.

10. Závěr

Systém byl otestován zadáváním nejrůznějších knihoven a schémat. Systém je funkční a je možné ho nasadit do reálného provozu, je však pravda, že je ještě spousta možností jak jej rozšířit. Například o logické porovnávání na základě funkčnosti, nyní je implementováno jen objektové porovnání. Vzhledem k omezení v implementované verzi knihovny JGraph by bylo možné uvažovat o přechodu na novější verzi knihovny, která je však značně odlišná od nyní použité. V novější verzi by bylo možné využívat jako předlohy k prvkům grafu vektorové formáty vstupů, což se přes nejrůznější pokusy nepodařilo v této verzi realizovat. Práce na této práci byla pro autora velice přínosná, narazil na spoustu zajímavostí jazyka Java či knihovny JGraph, ale také se blíže seznámil s webovým prostředím, skriptovacím jazykem PHP a databází MySQL.

Literatura

[1] HLÍNĚNÝ, Petr. *Diskrétní Matematika (456-533 DIM)*. : FEI VŠB-TUO, 2006. Dostupné z WWW: <http://homel.vsb.cz/~kov16/predmety_dm_1011.php>

[2] *JGraph User Manual : For JGraph Version 5.13.0.0*. Northampton : JGraph Ltd., 2009.

CD-ROM

Obsah přiloženého CD:

- Tento dokument
- Uživatelská příručka
- Zdrojové soubory appletů, včetně přeložených tříd
- PHP skripty pro webové rozhraní
- SQL příkaz pro vytvoření tabulek databáze
- Javadoc
- Datový model
- Běhové prostředí Javy 1.7